News Recommender System

Anonymous Alligators
April 4, 2022

Bhavik (MS18098), Aniket (MS18126), Kunal (MS18148), Akshay (MS18117)

1 Introduction

In this assignment, we aim to design, implement and analyze a news recommendation system for the company **Alligator's Den**. The objective is to increase retention of the customers by maximising engagement (through intelligent recommendation of articles), right from their first visit. The problem is divided into building two intelligent 'bots' - the article recommender and the user profiler. From the perspective of the app, we want to reduce over-customization to reduce bias and maximize coverage of the news corpus.

2 Data Scraping

We have used the following four websites to scrape the article data for this project: NDTV, IBTimes, India Today and El Pais. For each article, we collect a 6-tuple of meta-data like so:

(Headline, Published Date, URL, Category, Summary, Content)

These are then stored in an SQLite database for further pre-processing (cleaning, tokenization, stemming, etc) before they are vectorized for further usage.

3 Extracting article features

The corpus will be vectorized using TF-IDF and fed into an LDA (Latent-Dirichlet-Allocation) model to extract an optimal number of latent topics (list of words) that we will use later as a metric for document similarity. We also have thoughts of using Named-Entity-Recognition (to obtain contextual keywords) as a secondary filter to utilize in addition to preliminary similarity comparisons.

4 User Profiles

In order to make recommendations, we will have to gather session data (links clicked, session duration, time spent on an article, amount scrolled, etc) to extract information that is aggregated to calculate an 'engagement score' ($e \in [0, 1]$) for each document that a user visits.

4.1 Engagement Scores

Loosely, we plan to utilize a three-fold scheme to generate an engagement score:

- the article scroll-percentage is uniformly mapped to a rating (if the article is long enough to be scrolled).
- the user's reading time is weighted against a normal distribution around a mean value for each article that provides a rating which drops off piece-wise exponentially around the mean reading time.
- if the user reaches the end of an article, an (optional) explicit binary rating system is used (thumbs up/down) to gather user-preferences directly which is given a much larger weight.

In principle, the weights for each of the above contributions must be dynamically learnt/tweaked as new user ratings are gathered (and compared with our models' predicted ratings for an article). But we plan to hard-code them as hyper-parameters for the time being.

4.2 Building the profile

As part of the click-stream data, we possess a list of all documents (d_i) that a user has visited, as well the duration of time elapsed between visits (Δt). A user is simply taken to be a weighted average of the documents he has visited, with the weights being the engagement scores, and an exponential drop-off with session-time to take into account their evolving preferences:

$$u_i = \sum_i e_i \cdot d_i \cdot e^{-\lambda \Delta t}$$

The document vectors, d_i used here are not a bag of words, but rather a bag of topics (percentage constitution of the topics retrieved by LDA) and the user is simply an 'average document' of sorts. The idea is that, taking into account the topics rather than words will retain more context about user preferences upon averaging.

4.3 Tackling cold-start

- Upon registration and first login, a prompt of 8-10 prominent news categories is presented that the user chooses from (knowledge-based recommendation). This data is then used to present a specific section of the corpus (expected to be more similar to the user's tastes).
- We also intend to use some simple mathematical models to simulate users as a means of generating click-stream data and a user database to facilitate collaborative filtering.

5 Recommending articles

We will utilize a hybrid approach of content-based and collaborative filtering to recommend articles.

- **Content-based filtering:** The user profile is itself an 'average' article of the user's history, so we simply calculate cosine similarity with other articles (in their topic constituent form) to get top ten 'nearest' articles (and filter out those that have already been visited). Locality sensitive hashing (LSH) can be used to reduce time complexity of these comparisons.
- **Collaborative filtering:** Once the user has visited a sizeable number of articles, we can construct an incomplete user-article matrix of engagement scores, and learn the missing scores (by means of optimizing through matrix factorization). The predicted scores can then be used to recommend the top ten 'highest-rated' articles (as predicted by the model).

We combine the two previous lists and apply an extra exponential factor based on published date (to factor in the 'freshness' of news), to re-order the articles in terms of predicted engagement and obtain the final list of recommendations.

6 Evaluating the system

In order to make sure that our system actually works and is not simply recommending random articles, we must utilize a simple evaluation scheme for the system. This can be done by setting a threshold and comparing the predicted rating (engagement) for an article, with the observed rating (once the user has consumed the recommended article) within this threshold to make a binary classification of the goodness of recommendation. This could optionally be used as a cost function to learn new weights for calculation of engagement scores, or any other hyper-parameters that characterize the recommendation system.
